

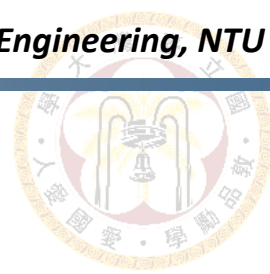
Computer-Aided VLSI System Design

Homework 3: Simple Convolution and Image Processing Engine

Graduate Institute of Electronics Engineering, National Taiwan University

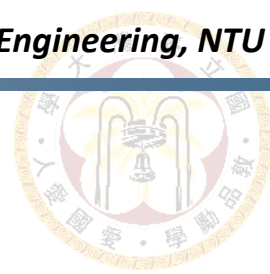


NTU GIEE



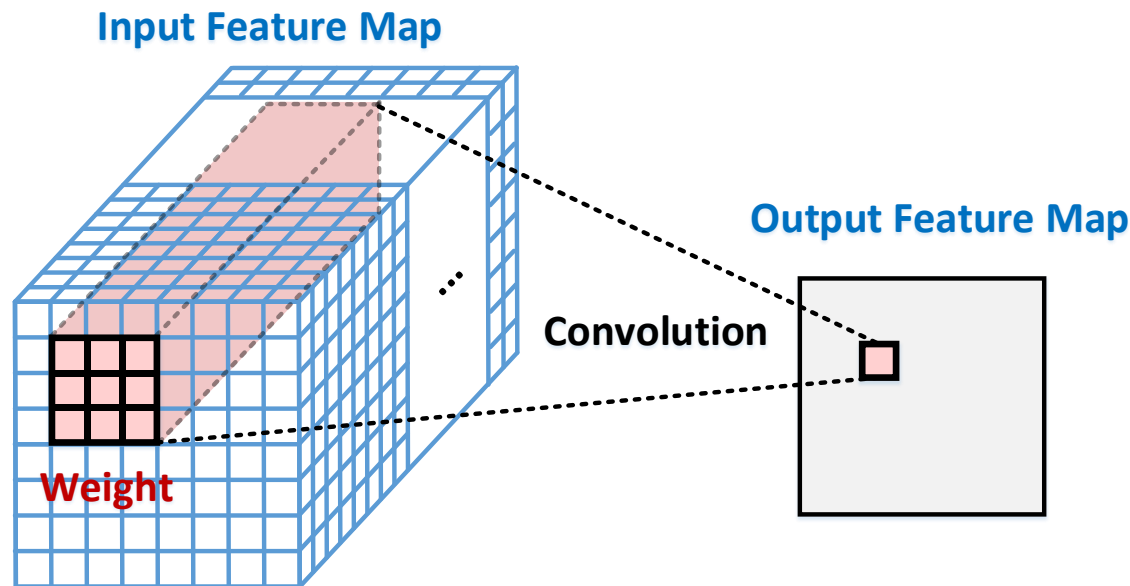
Goal

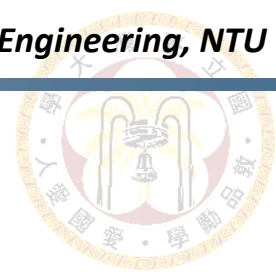
- In this homework, you will learn
 - How to synthesis your design
 - How to run gate-level simulation
 - How to use SRAM



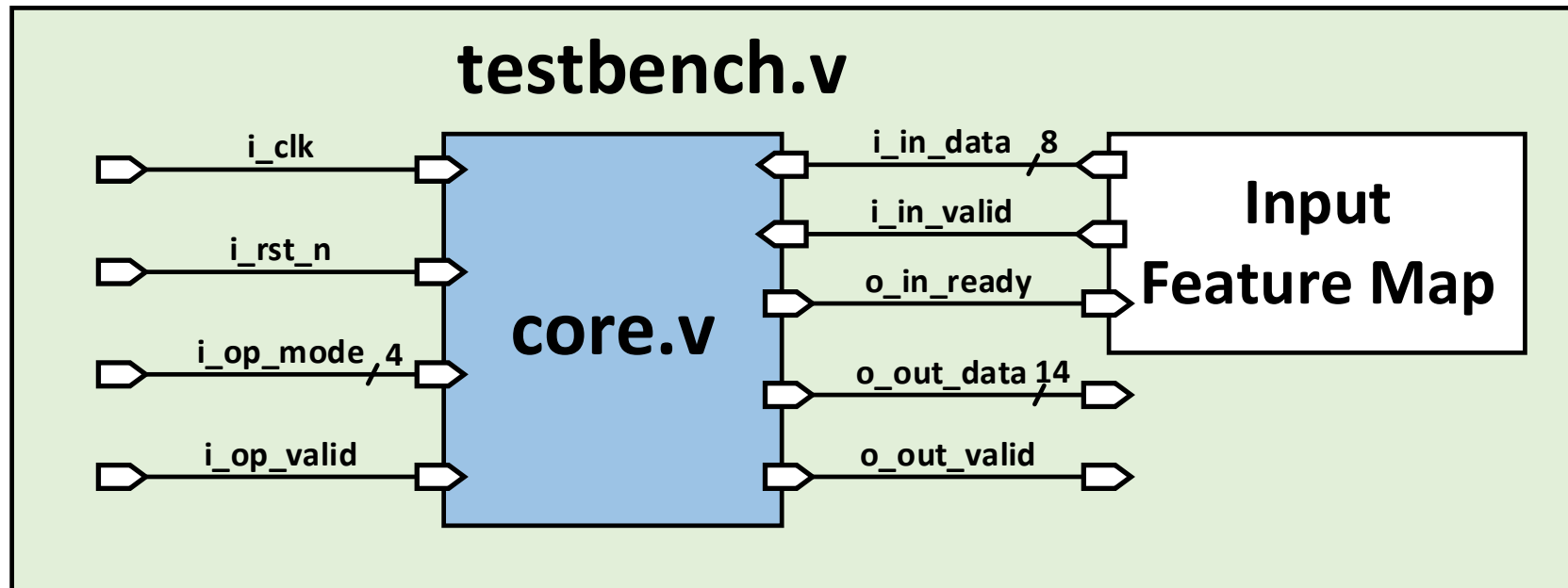
Introduction

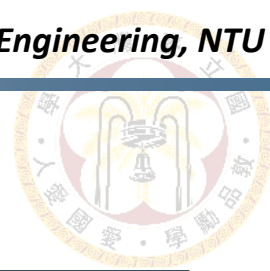
- In this homework, you are going to implement a simplified convolution and image processing engine. An $8 \times 8 \times 32$ feature map will be loaded first, and it will be processed with several functions.





Block Diagram





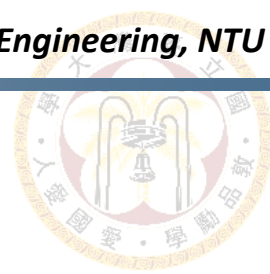
Input/Output

Signal Name	I/O	Width	Simple Description
i_clk	I	1	Clock signal in the system.
i_rst_n	I	1	Active low asynchronous reset.
i_op_valid	I	1	This signal is high if operation mode is valid
i_op_mode	I	4	Operation mode for processing
o_op_ready	O	1	Set high if ready to get next operation
i_in_valid	I	1	This signal is high if input pixel data is valid
i_in_data	I	8	Input pixel data (unsigned)
o_in_ready	O	1	Set high if ready to get next input data (only valid for i_op_mode = 4'b0000)
o_out_valid	O	1	Set high with valid output data
o_out_data	O	14	Pixel data or image processing result (signed)



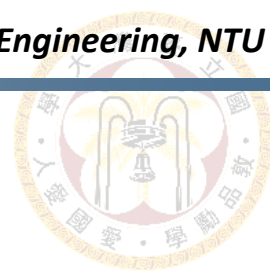
Specification(1)

- All inputs are synchronized with the **negative** edge clock
- All outputs should be synchronized at clock **rising** edge
- You should reset all your outputs when i_rst_n is **low**
 - Active low asynchronous reset is used and only once



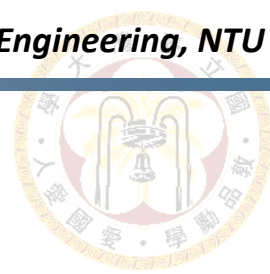
Specification(2)

- Operations are given by i_op_mode when i_op_valid is **high**
- i_op_valid stays only **1** cycle
- i_in_valid and o_out_valid can't be **high** in the same time
- i_op_valid and o_out_valid can't be **high** in the same time
- i_in_valid and o_op_ready can't be **high** in the same time
- i_op_valid and o_op_ready can't be **high** in the same time
- o_op_ready and o_out_valid can't be **high** in the same time



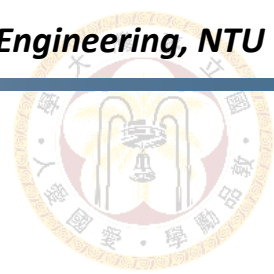
Specification(3)

- Set o_op_ready to **high** to get next operation (only one cycle)
 - Raise o_op_ready only when the design is prepared for the next operation
- o_out_valid should be **high** for valid output results
- **At least one SRAM** is implemented in your design



Specification(4)

- Only worst-case library is used for synthesis.
- The synthesis result of data type should **NOT** include any **Latch**.
- The slack for setup-time should be **non-negative**.
- **No any timing violation and glitches** for the gate level simulation **after reset**.

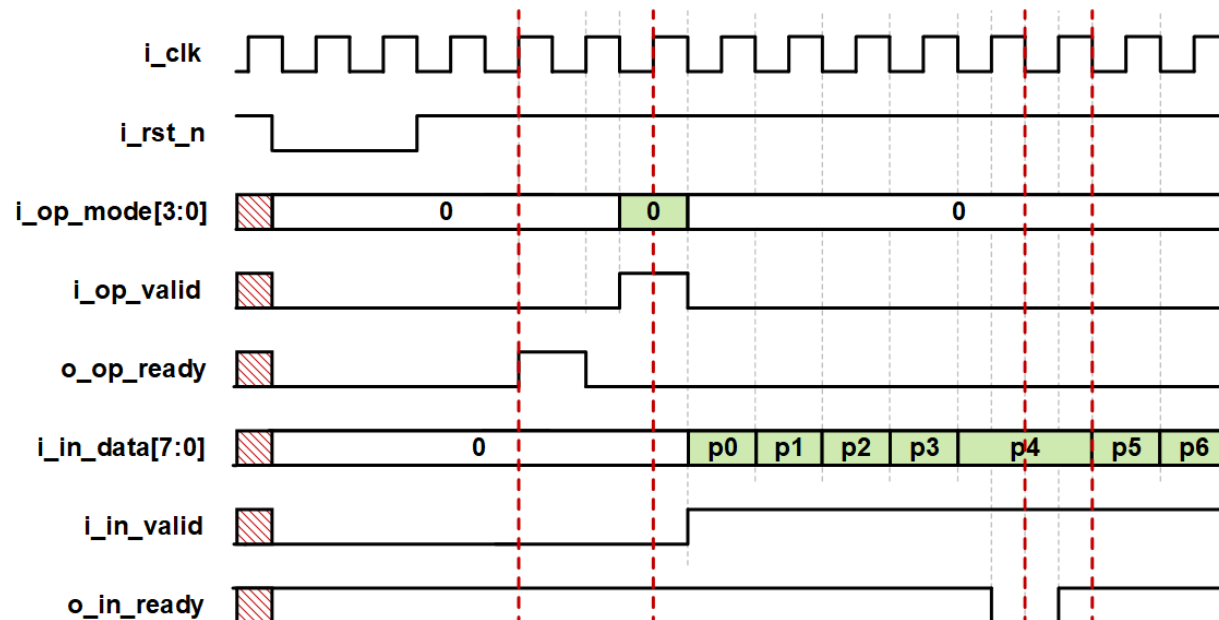


Waveform: Loading Image

$o_op_ready = 0$ is raised to 1 to get the first operation

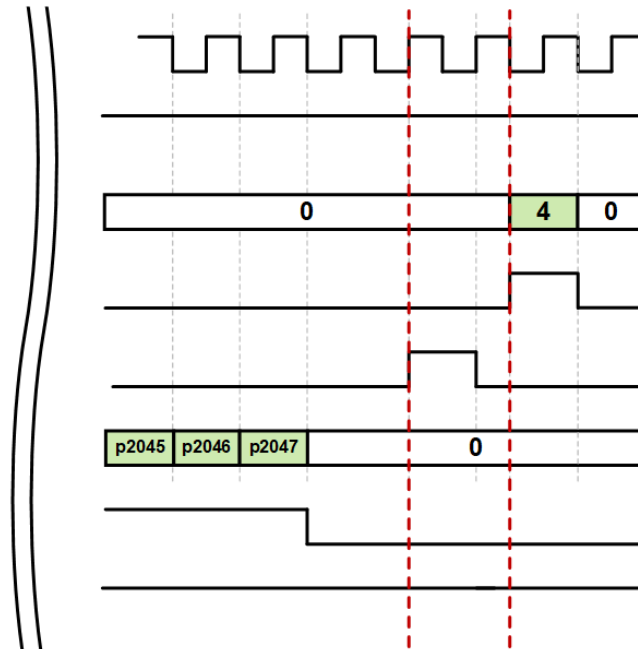
Testbench detects $o_in_ready = 1$ at negative edge, resume streaming input

i_op_mode & i_op_valid will be next operation at the falling edge for one cycle

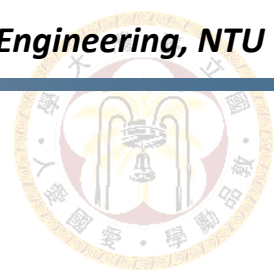


$i_op_mode = 0$, start loading image at the next rising edge

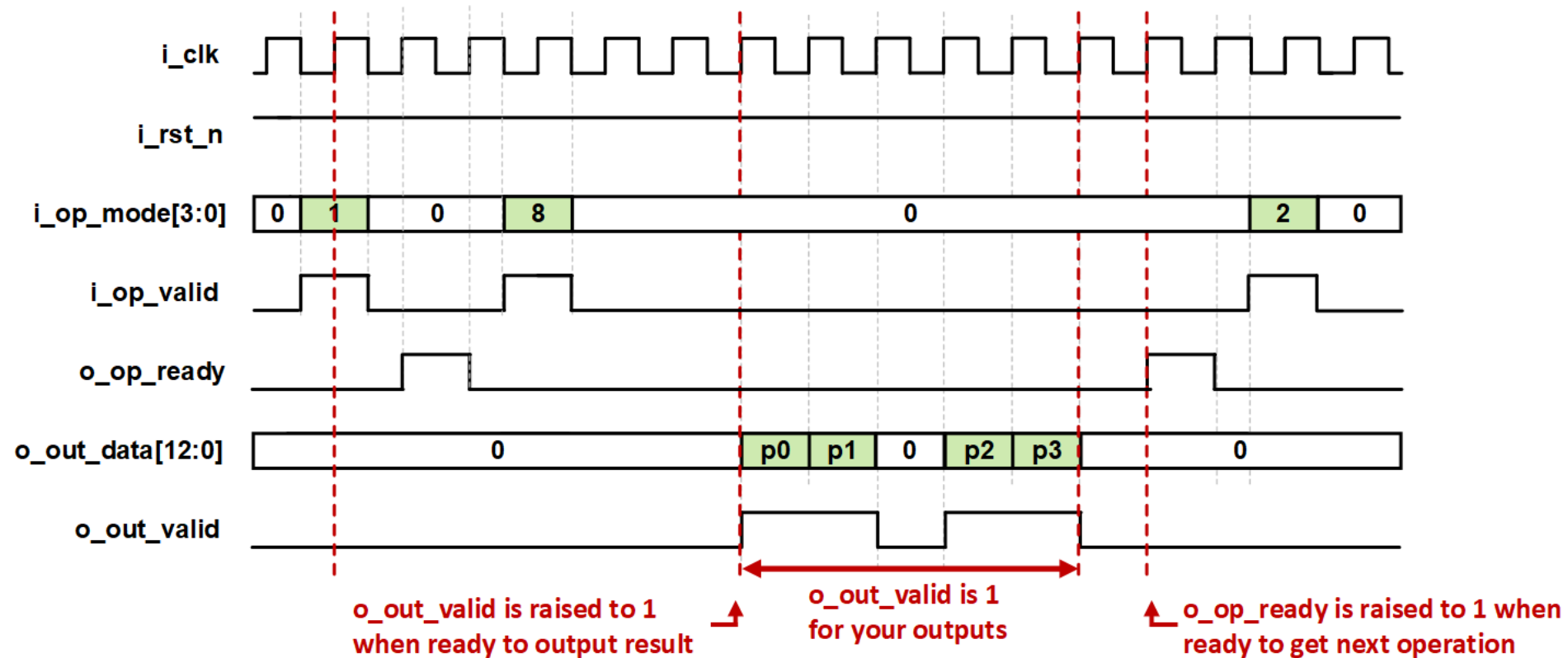
Testbench detects $o_in_ready = 0$ at negative edge, stop streaming input

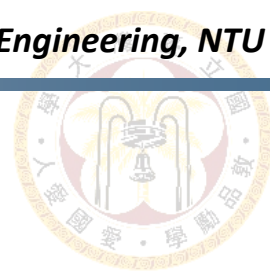


o_op_ready is raised to 1 after loading whole image



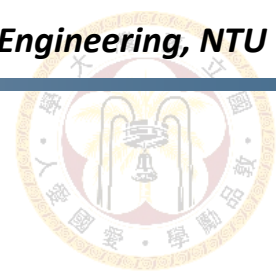
Waveform: Other Operations





Operation Modes

i_op_mode	Meaning
4'b0000	Input feature map loading
4'b0001	Origin right shift
4'b0010	Origin left shift
4'b0011	Origin up shift
4'b0100	Origin down shift
4'b0101	Reduce the channel depth of the display region
4'b0110	Increase the channel depth of the display region
4'b0111	Output the pixels in the display region
4'b1000	Perform convolution in the display region
4'b1001	Median filter operation
4'b1010	Sobel gradient + non-maximum suppression (NMS)



Input Image

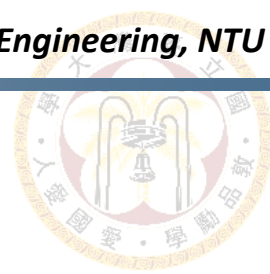
- The input image is given in **raster-scan** order

channel 31								1984	1985	1986	1987	1988	1989	1990	1991			
								1992	1993	1994	1995	1996	1997	1998	1999			
channel 1								64	65	66	67	68	69	70	71	2006	2007	
																2014	2015	
channel 0								0	1	2	3	4	5	6	7	79	2022	2023
								8	9	10	11	12	13	14	15	87	2030	2031
								16	17	18	19	20	21	22	23	95	2038	2039
								24	25	26	27	28	29	30	31	103	2046	2047
								32	33	34	35	36	37	38	39	111		
								40	41	42	43	44	45	46	47	119		
								48	49	50	51	52	53	54	55	127		
								56	57	58	59	60	61	62	63			



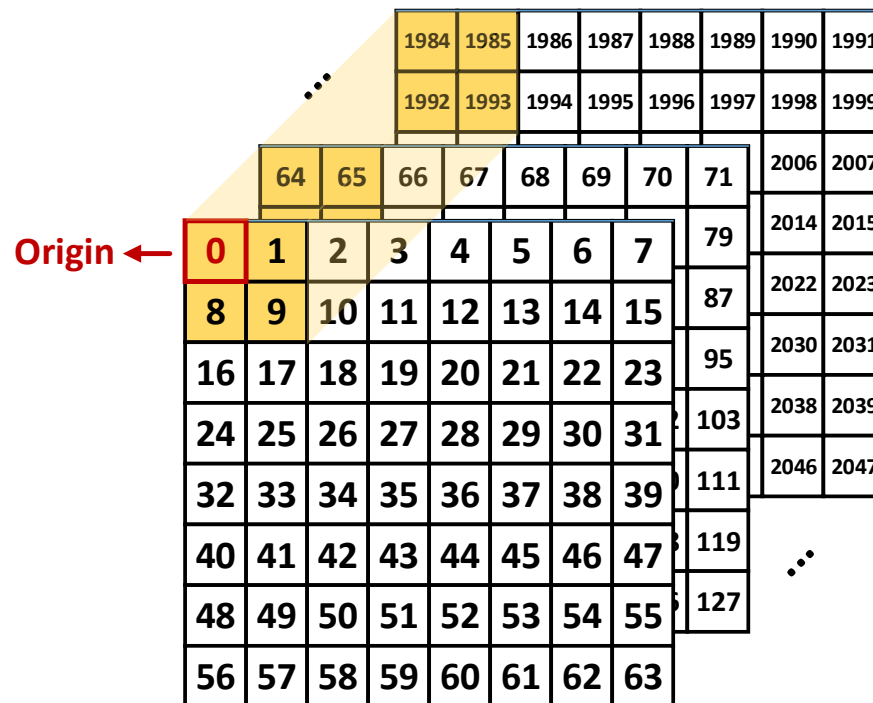
Input Image Loading

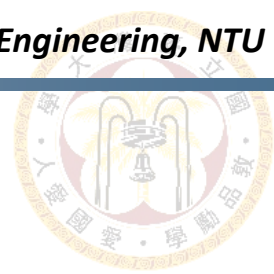
- An 8x8x32 image is loaded for 2048 cycles in **raster-scan** order
- The size of each pixel is 8 bits (unsigned)
- Raise **o_op_ready** to 1 after loading all image pixels
- If **o_in_ready** is 0, stop input data until **o_in_ready** is 1
- The input feature map will be loaded only once at the beginning



Origin

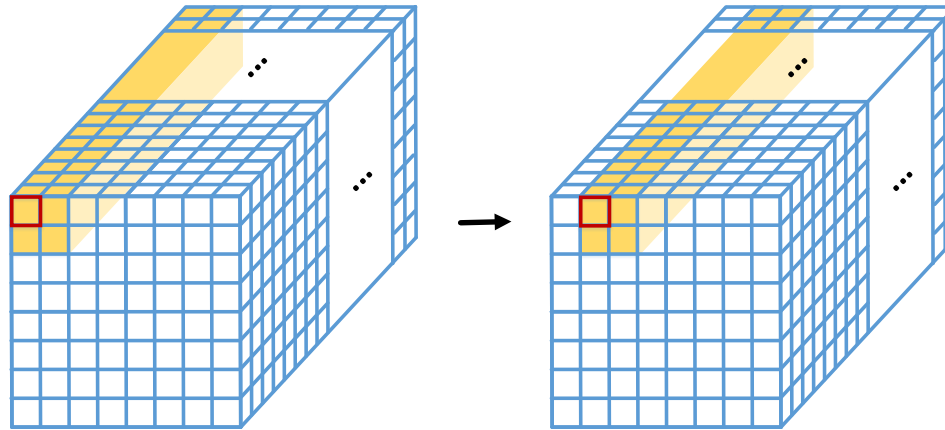
- The first pixel in the display region is **origin**
- The default coordinate of the origin is at 0
- The size of the display region is $2 \times 2 \times \text{depth}$



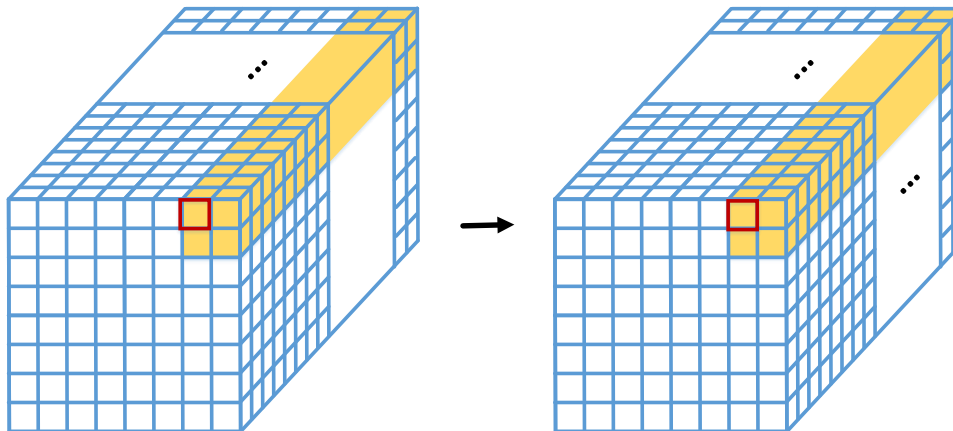


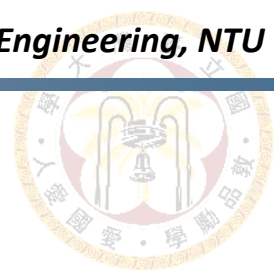
Origin Shifting

- Origin right shift



- If output of display exceeds the boundary, retain the same origin

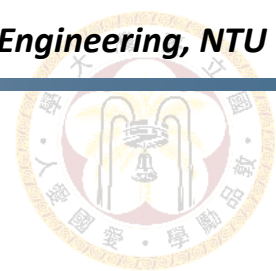




Channel Depth

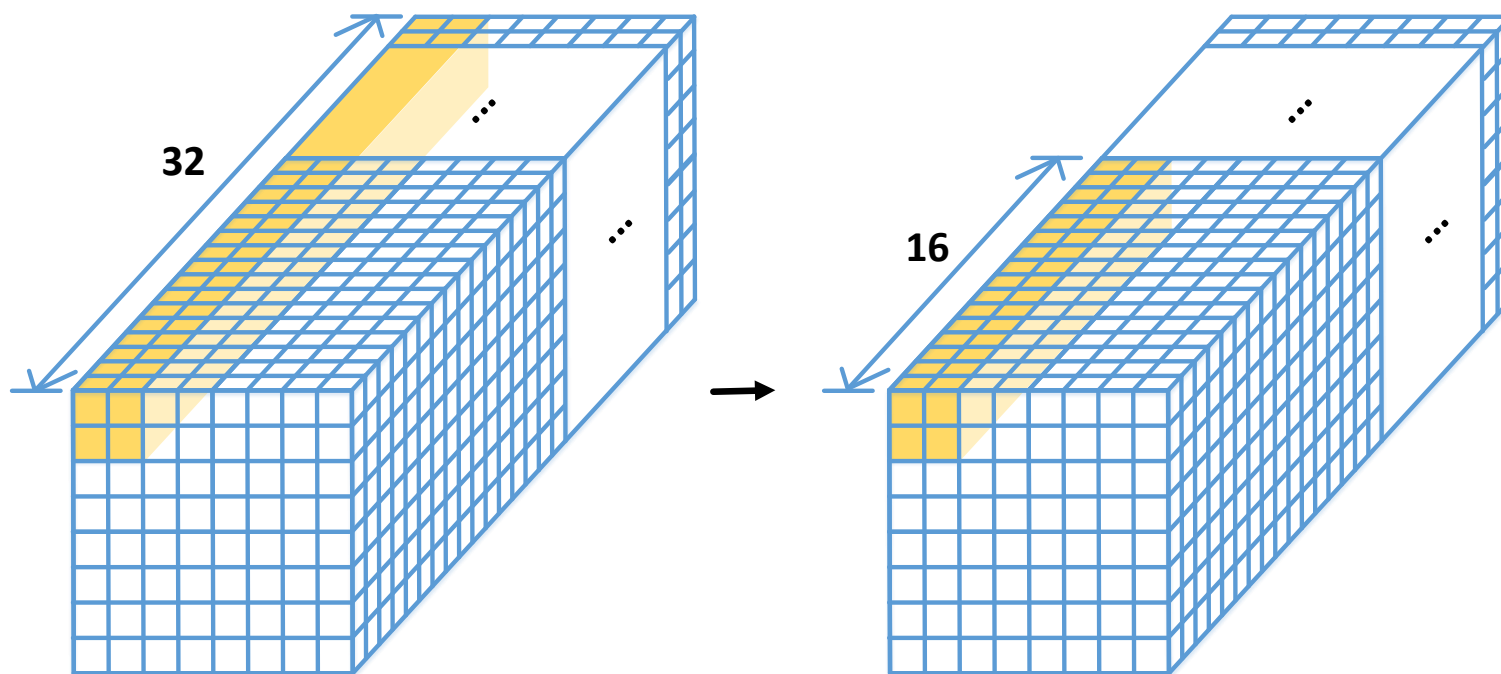
- 3 depths are considered in this design
 - 32, 16, 8
 - Default depth is 32
- The display size will change according to different depth

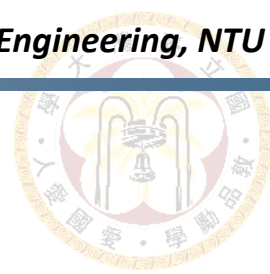
Depth	Display size
32	2 x 2 x 32
16	2 x 2 x 16
8	2 x 2 x 8



Scale-down

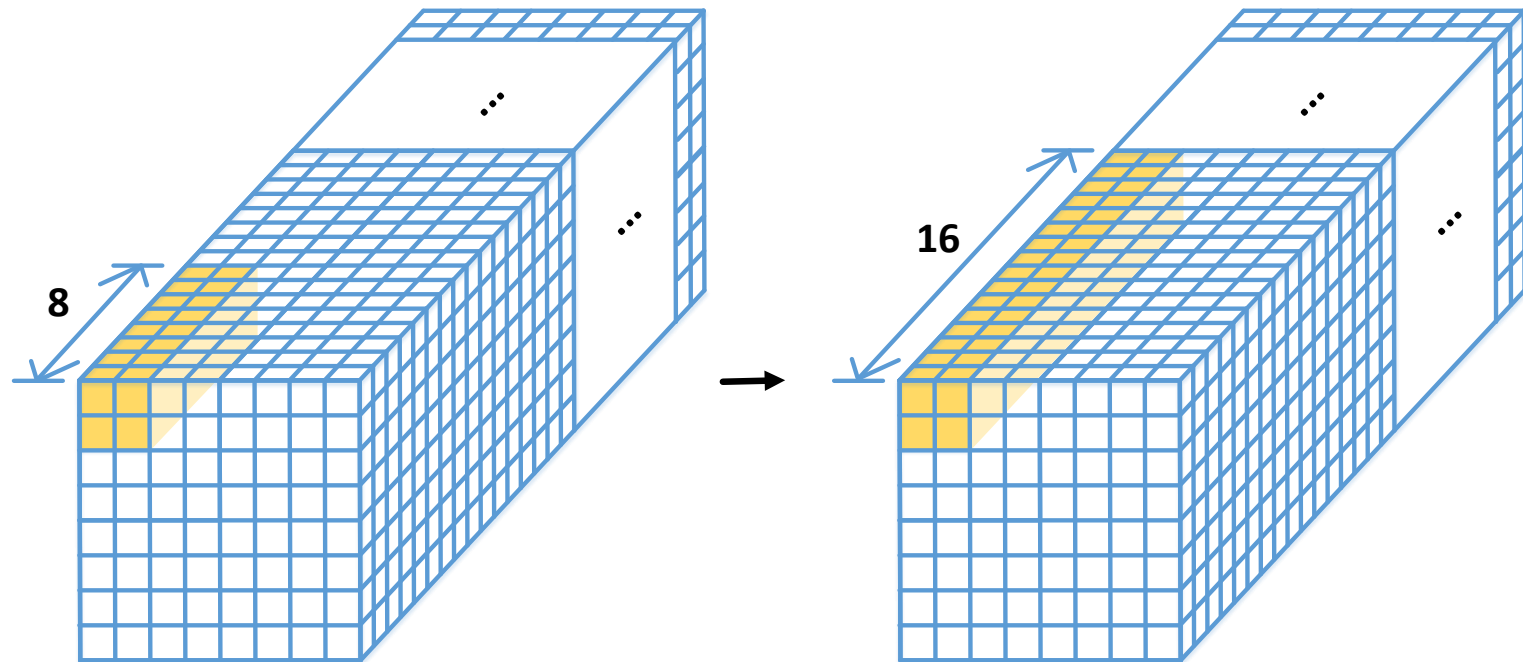
- Reduce the channel depth of the display region
 - Ex. For channel depth, $32 \rightarrow 16 \rightarrow 8$
- If the depth is 8, retain the same depth

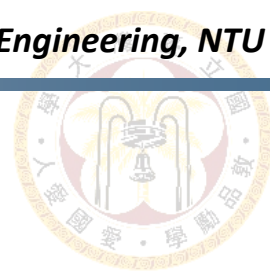




Scale-up

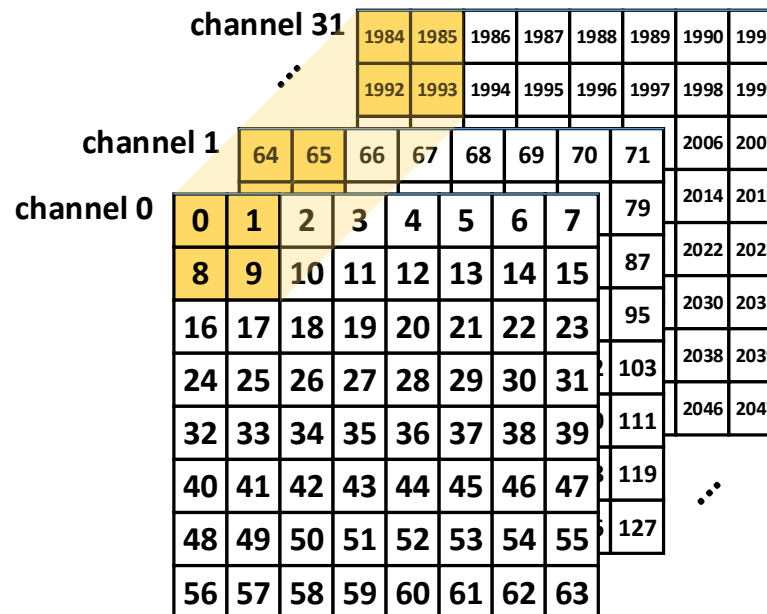
- Increase the channel depth of the display region
 - Ex. For channel depth, $8 \rightarrow 16 \rightarrow 32$
- If the depth is 32, retain the same depth

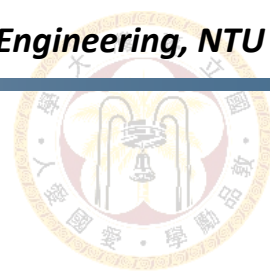




Display

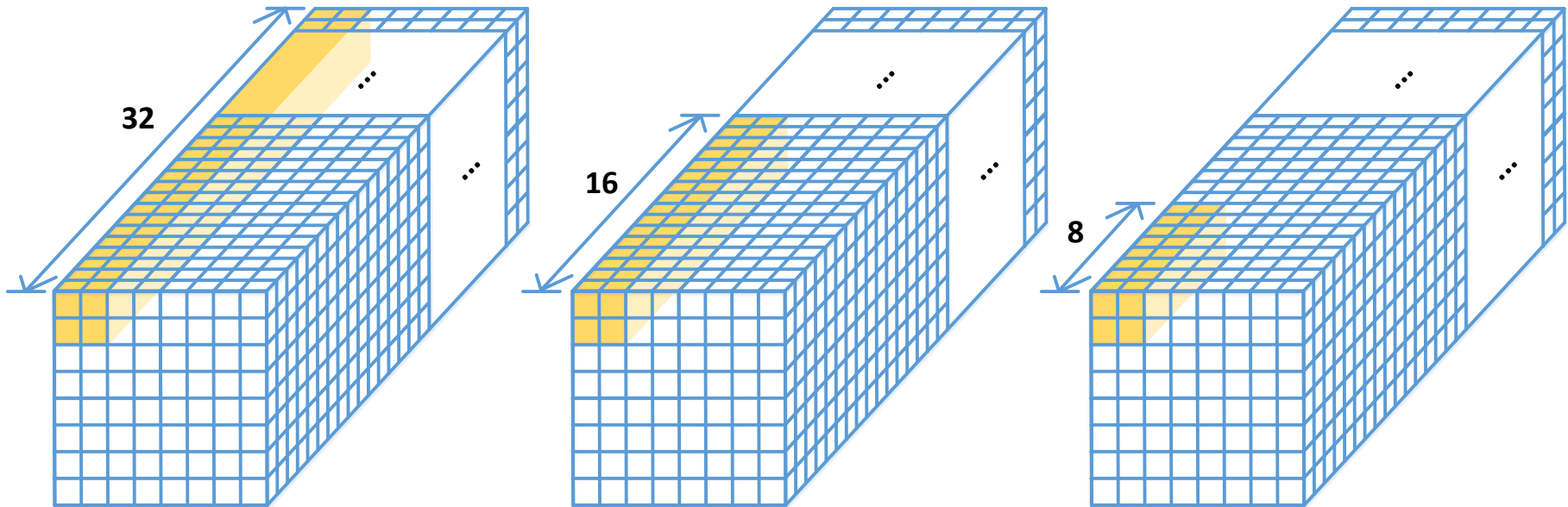
- You have to output the pixels in the display region
- Set **o_out_data [13:8]** to 0 and **o_out_data [7:0]** to pixel data
- The pixels are displayed in **raster-scan** order
 - For example: 0 → 1 → 8 → 9 → 64 → 65 → ... → 1992 → 1993





Display

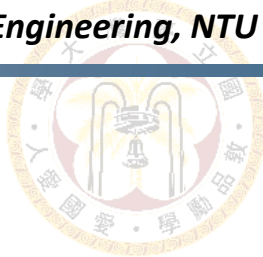
- For display, the display size changes according to the depth



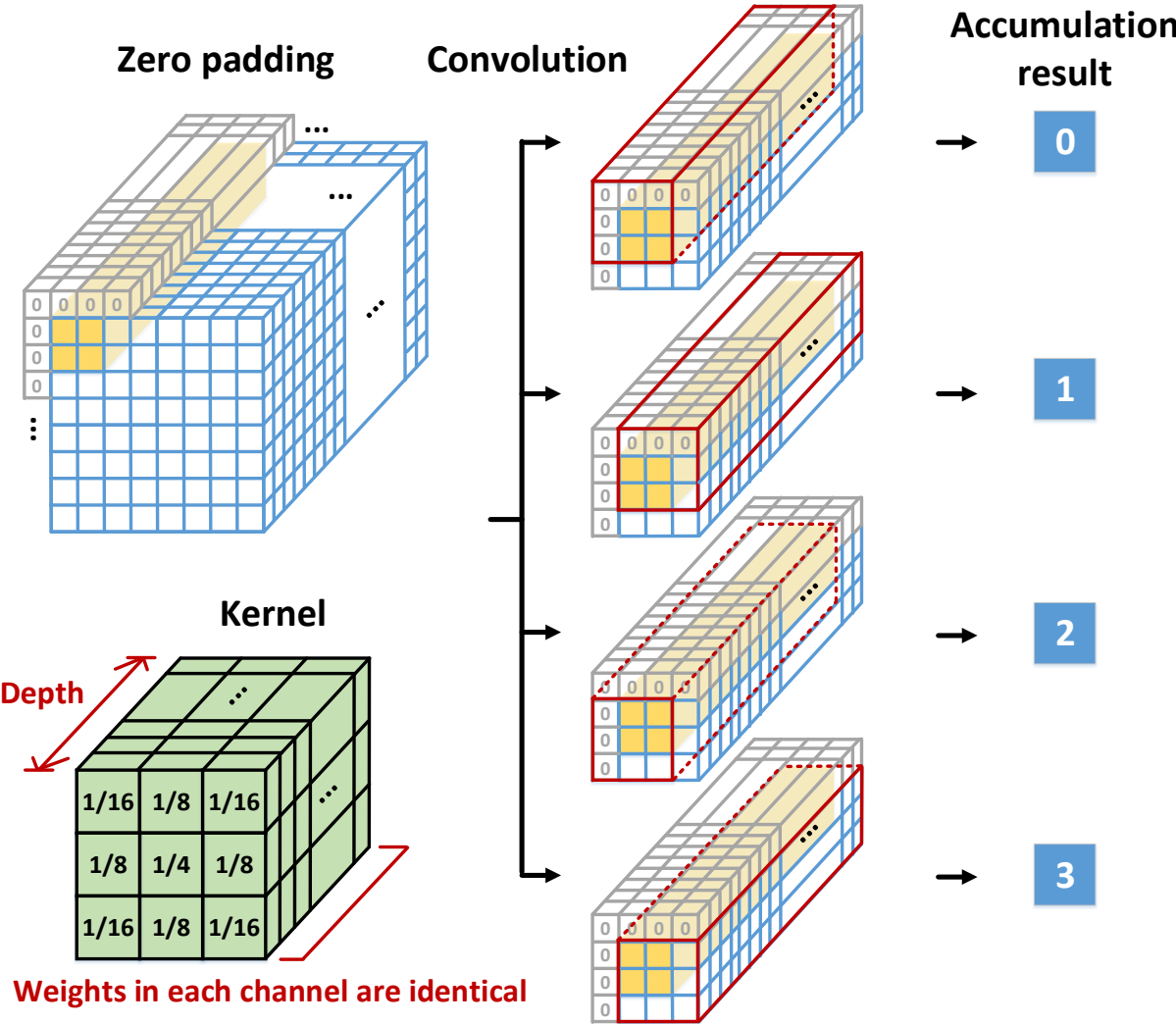


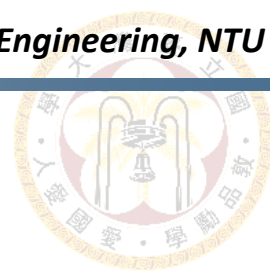
Convolution

- For this operation, you have to perform convolution **in the display region**
- The size of the kernel is a $3 \times 3 \times \text{depth}$ and the weights in each channel are identical
- The feature map needs to be zero-padded for convolution
- The accumulation results should be **rounded to the nearest integer** [1]
 - Do not truncate temporary results during computation
- After the convolution, you have to output the **4** accumulation results in **raster-scan** order
- The values of original pixels will not be changed



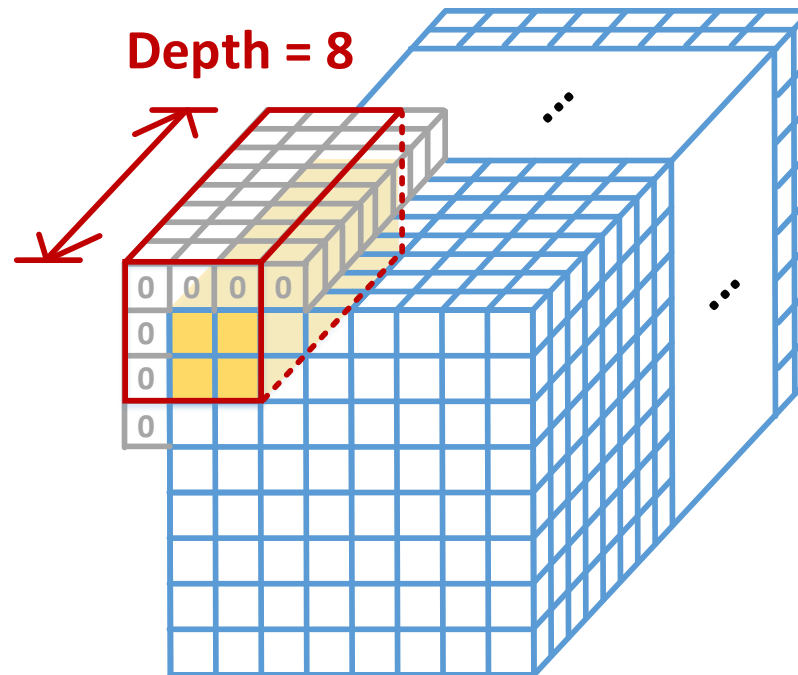
Example of Convolution

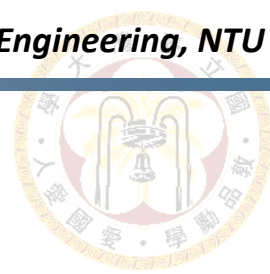




Example of Convolution

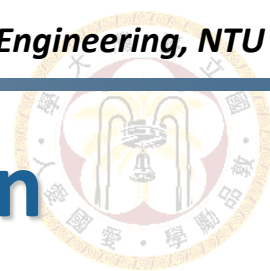
- The number of channels that are accumulated during convolution is determined by the depth.
 - For example, accumulate 8 channels if the depth is 8.



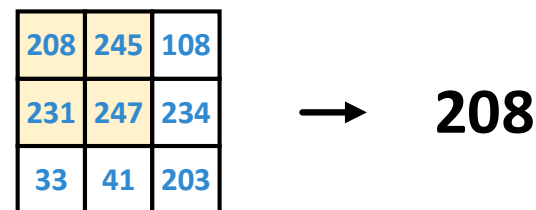
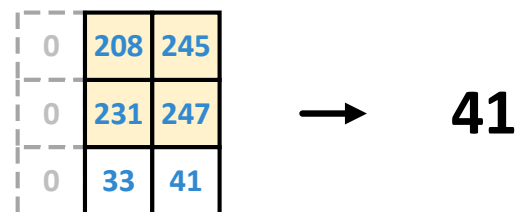
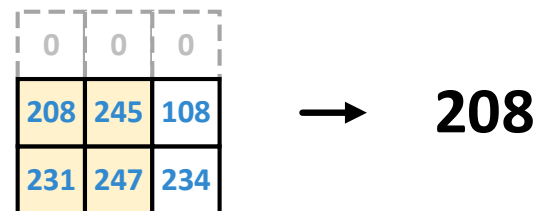
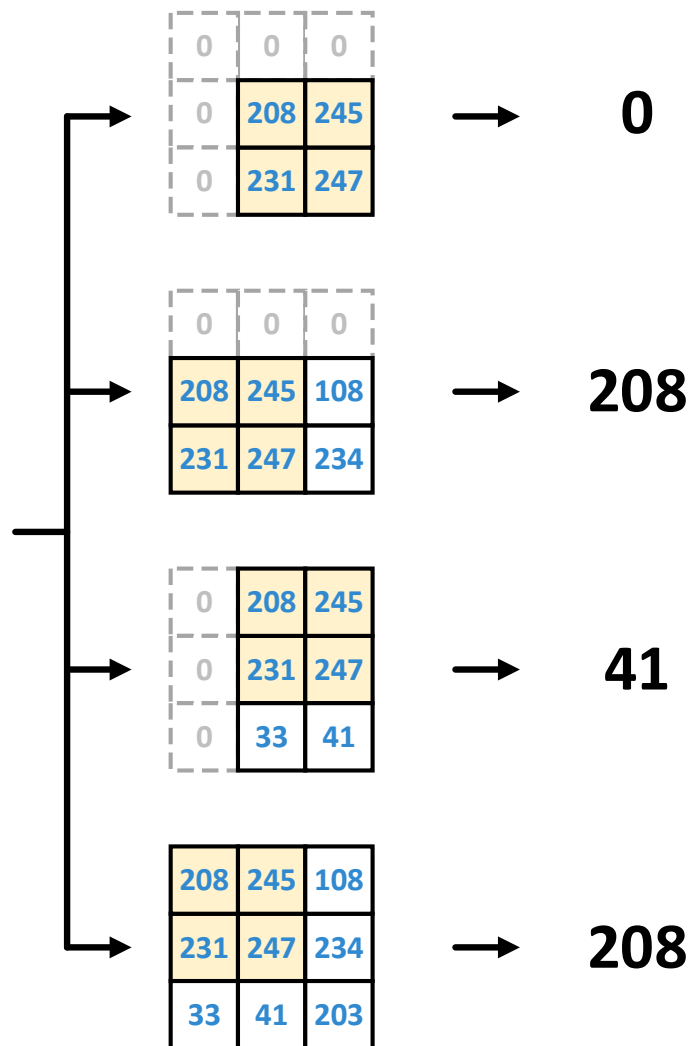
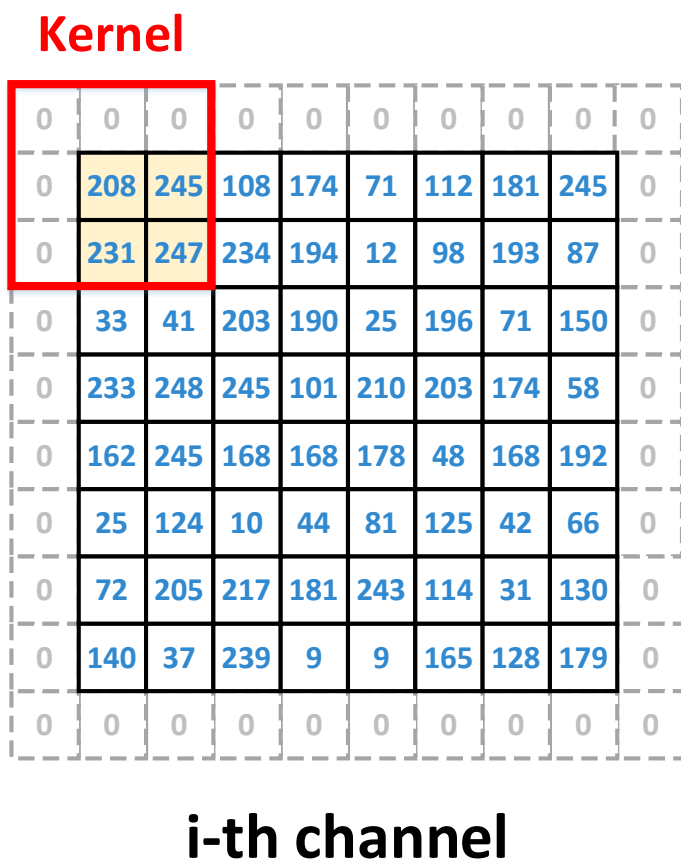


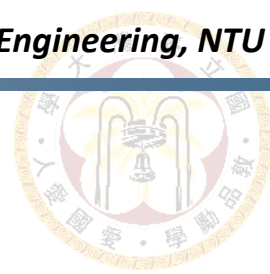
Median Filter Operation

- For this operation, you have to perform median filtering **in the first 4 channels of the display region**
- The kernel size of the median filter is 3×3
- Perform median filtering on each channel **separately**
- The feature map needs to be zero-padded for median filter operation
- After median filtering, you have to output the **$2 \times 2 \times 4$** filtered results in **raster-scan** order
 - Set **o_out_data [13:8]** to 0 and **o_out_data [7:0]** to pixel data
- The values of original pixels will not be changed



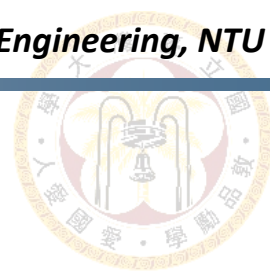
Example of Median Filter Operation





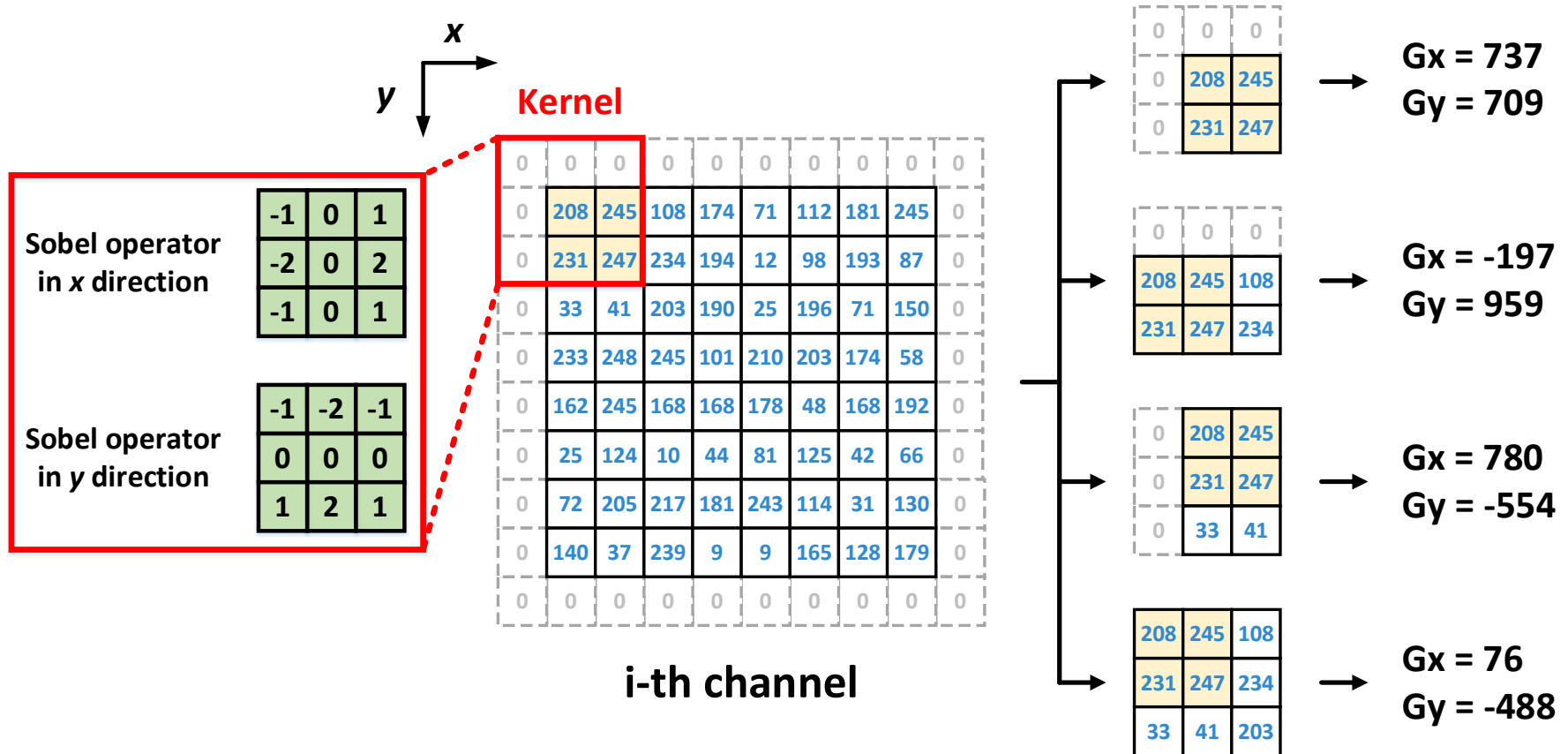
Sobel Gradient + NMS

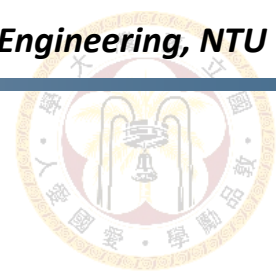
- Calculate gradient **in the first 4 channels of the display region** using the Sobel operator and retain only local maxima along the gradient direction
 - Conduct computations **separately** for each channel
- The kernel size of the Sobel operator is 3 x 3
- The feature map needs to be zero-padded
- After computation, you have to output the **2 x 2 x 4** results in **raster-scan** order
- The values of original pixels will not be changed



Sobel Gradient + NMS

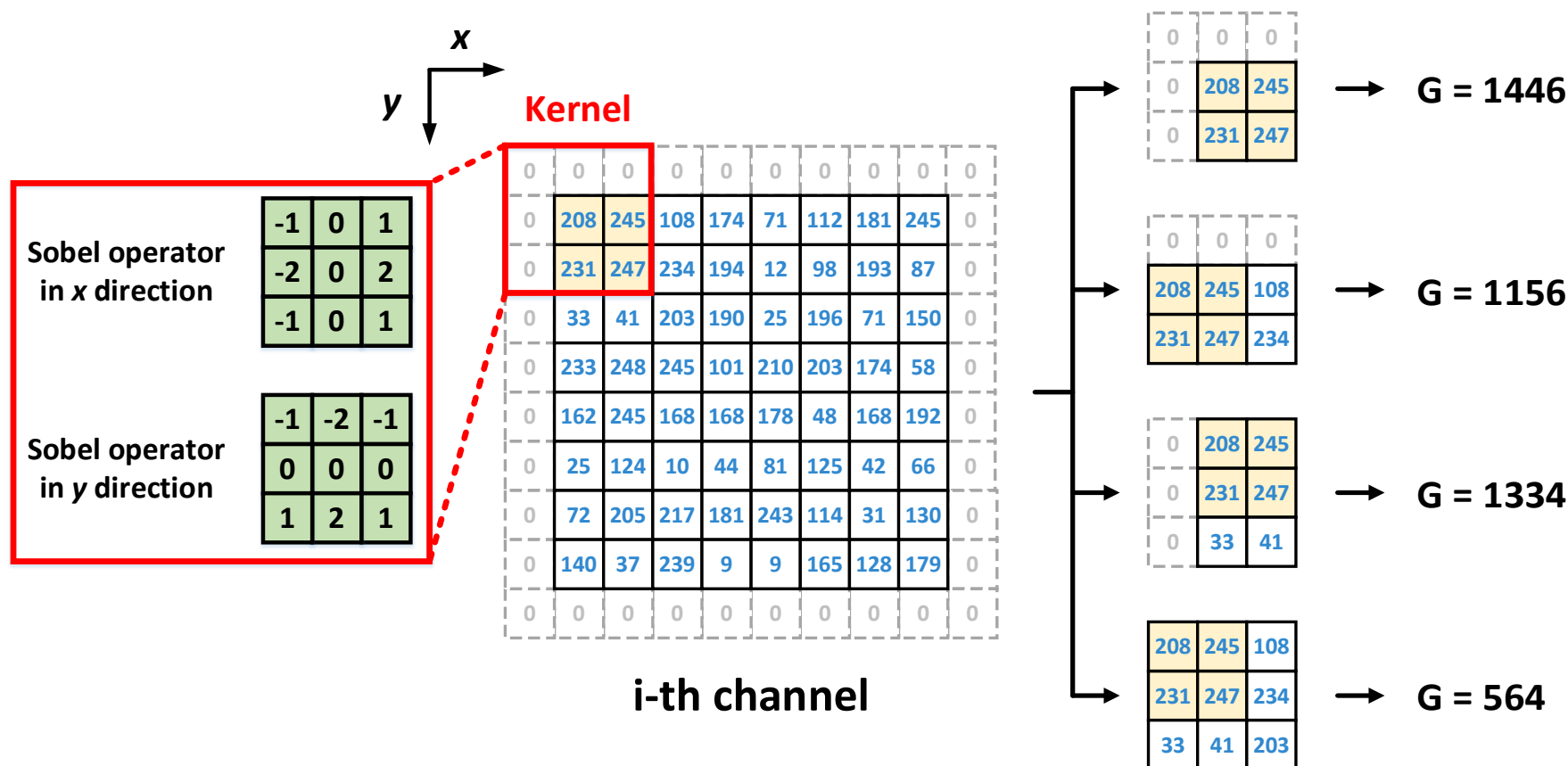
- Gradient magnitude $G(x, y) = |G_x(x, y)| + |G_y(x, y)|$





Sobel Gradient + NMS

- Gradient magnitude $G(x, y) = |G_x(x, y)| + |G_y(x, y)|$

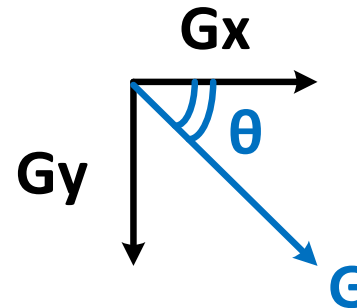




Sobel Gradient + NMS

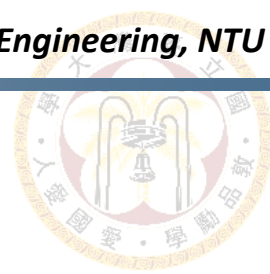
- Gradient direction

$$\theta(x, y) = \tan^{-1} \left(\frac{G_y(x, y)}{G_x(x, y)} \right)$$



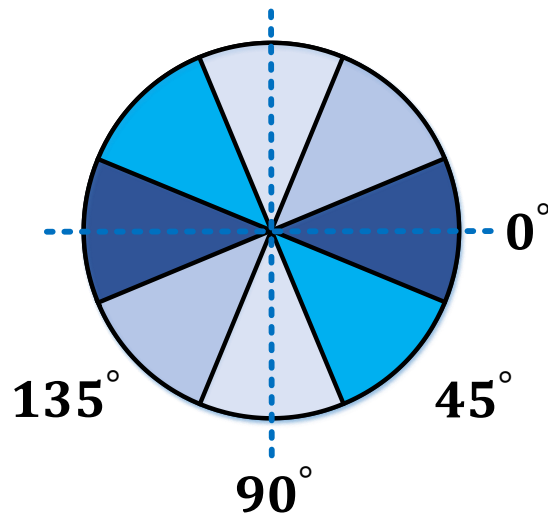
- Tangent approximation

Tangent	Approx. Value	Tangent	Approx. Value
$\tan 0^\circ$	0	$\tan 112.5^\circ$	$-\tan 67.5^\circ$
$\tan 22.5^\circ$	$2^{-2} + 2^{-3} + 2^{-5} + 2^{-7}$	$\tan 135^\circ$	$-\tan 45^\circ$
$\tan 45^\circ$	1	$\tan 157.5^\circ$	$-\tan 22.5^\circ$
$\tan 67.5^\circ$	$2 + 2^{-2} + 2^{-3} + 2^{-5} + 2^{-7}$	$\tan 180^\circ$	$-\tan 0^\circ$



Sobel Gradient + NMS

- Non-maximum suppression (NMS)
 - Find the direction $d_k \in \{0^\circ, 45^\circ, 90^\circ, 135^\circ\}$ that is closest to the gradient direction $\theta(x, y)$
 - If the value of $G(x, y)$ is less than any of its two neighbors along d_k , then set $G(x, y)$ to 0 (suppression)

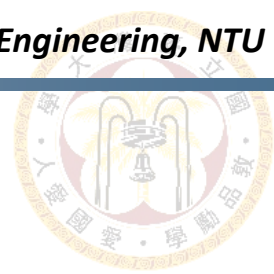


0	0	0	0
0	$G_{(0,0)}$	$G_{(1,0)}$	0
0	$G_{(0,1)}$	$G_{(1,1)}$	0
0	0	0	0

Example:

If $22.5^\circ \leq \theta(0,0) \leq 67.5^\circ$, then compare $G(0,0)$ with its two neighbors along 45° direction, i. e. 0 and $G(1,1)$.

If $112.5^\circ \leq \theta(0,1) \leq 157.5^\circ$, then compare $G(0,1)$ with its two neighbors along 135° direction, i. e. 0 and $G(1,0)$.



Testbench

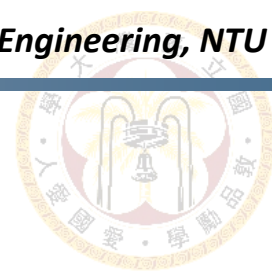
```
`timescale 1ns/100ps
`define CYCLE      10.0    // CLK period.
`define HCYCLE     (`CYCLE/2)
`define MAX_CYCLE  1000000
`define RST_DELAY  2

`ifdef tb1
  `define INFILE "../00_TESTBED/PATTERN/indata1.dat"
  `define OPFILE "../00_TESTBED/PATTERN/opmode1.dat"
  `define GOLDEN "../00_TESTBED/PATTERN/golden1.dat"
`elseif tb2
  `define INFILE "../00_TESTBED/PATTERN/indata2.dat"
  `define OPFILE "../00_TESTBED/PATTERN/opmode2.dat"
  `define GOLDEN "../00_TESTBED/PATTERN/golden2.dat"
`elseif tb3
  `define INFILE "../00_TESTBED/PATTERN/indata3.dat"
  `define OPFILE "../00_TESTBED/PATTERN/opmode3.dat"
  `define GOLDEN "../00_TESTBED/PATTERN/golden3.dat"
```

```
`elseif tb4
  `define INFILE "../00_TESTBED/PATTERN/indata4.dat"
  `define OPFILE "../00_TESTBED/PATTERN/opmode4.dat"
  `define GOLDEN "../00_TESTBED/PATTERN/golden4.dat"
`elseif tbh
  `define INFILE "../00_TESTBED/PATTERN/indatah.dat"
  `define OPFILE "../00_TESTBED/PATTERN/opmodeh.dat"
  `define GOLDEN "../00_TESTBED/PATTERN/goldenh.dat"
`else
  `define INFILE "../00_TESTBED/PATTERN/indata0.dat"
  `define OPFILE "../00_TESTBED/PATTERN/opmode0.dat"
  `define GOLDEN "../00_TESTBED/PATTERN/golden0.dat"
`endif

`define SDFFILE "core_syn.sdf" // Modify your sdf file name
```

```
// For gate-level simulation only
`ifdef SDF
  initial $sdf_annotate(`SDFFILE, u_core);
  initial #1 $display("SDF File %s were used for this simulation.", `SDFFILE);
`endif
```

Pattern

indata*.dat

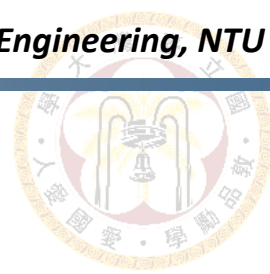
```
11010000
10101101
10001011
01010110
00101110
11110001
11110110
11001101
01000110
11001110
00001011
10101111
10001000
00111111
00001100
11110100
11100100
00101100
11000100
11001000
```

opmode*.dat

```
0000
0010
1000
0100
1000
0011
1000
0100
1000
0110
1000
0001
1000
0110
1000
0110
1000
0011
1000
0100
```

golden*.dat

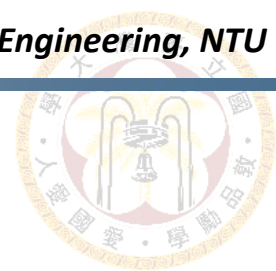
```
00100111111001
00110011000100
00110001011010
01000001010101
00110001011010
01000001010101
00101110110101
00111111110110
00100111111001
00110011000100
00110001011010
01000001010101
00110001011010
01000001010101
00101110110101
00111111110110
00110001011010
01000001010101
00101110110101
00111111110110
```



01_RTL

■ core.v

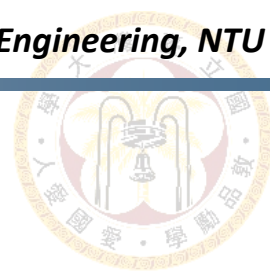
```
module core (                                //Don't modify interface
    input      i_clk,
    input      i_rst_n,
    input      i_op_valid,
    input [ 3:0] i_op_mode,
    output     o_op_ready,
    input      i_in_valid,
    input [ 7:0] i_in_data,
    output     o_in_ready,
    output     o_out_valid,
    output [13:0] o_out_data
);
```



01_RTL

■ rtl_01.f

```
// -----  
// Simulation: HW3  
// -----  
  
// testbench  
// -----  
../00_TESTBED/testbench.v  
  
// memory file  
// -----  
../sram_256x8/sram_256x8.v  
../sram_512x8/sram_512x8.v  
../sram_4096x8/sram_4096x8.v  
  
// design files  
// -----  
./core.v
```



01_RTL

- Run the RTL simulation under 01_RTL folder

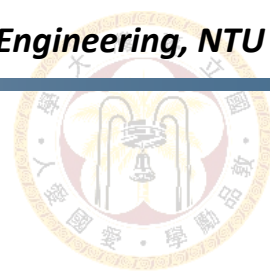
```
vcs -f rtl_01.f -full64 -R -debug_access+all +v2k  
+notimingcheck -sverilog +define+tb0
```

tb0, tb1, tb2, tb3, tb4

or

```
./01_run tb0 5.0
```

clock period



01_RTL

- SpyGlass linting

```
./02_lint
```

- Command for cleaning temporary files

```
./99_clean_up
```

- Note that before executing the shell script, change the file permissions by

```
chmod +x ./01_run ./02_lint ./99_clean_up
```



02_SYN

- **core_dc.sdc**

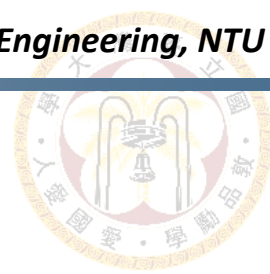
```
# operating conditions and boundary conditions #  
set cycle 5.0; # modify your clock cycle here #
```

- **flist.sv**

```
1 // list all paths to your design files  
2 `include "../01_RTL/core.v"
```

- Run the command to do synthesis

```
dc_shell-t -f syn.tcl | tee syn.log
```



03_GATE

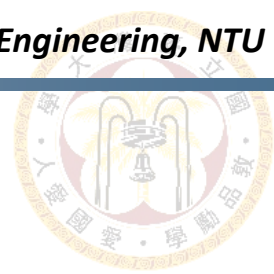
- Run gate-level simulation under 03_GATE folder

```
vcs -f rtl_03.f -full64 -R -debug_access+all +v2k  
+maxdelays -negdelay +neg_tchk +define+SDF+tb0
```

or

```
./03_run tb0 5.0
```

clock period



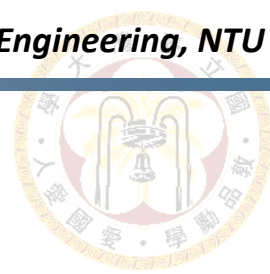
sram_256x8

Pin Description

Pin	Description
A[7:0]	Addresses (A[0] = LSB)
D[7:0]	Data Inputs (D[0] = LSB)
CLK	Clock Input
CEN	Chip Enable
WEN	Write Enable
Q[7:0]	Data Outputs (Q[0] = LSB)

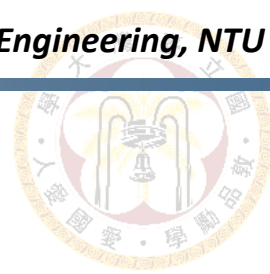
SRAM Logic Table

CEN	WEN	Data Out	Mode	Function
H	X	Last Data	Standby	Address inputs are disabled; data stored in the memory is retained, but the memory cannot be accessed for new reads or writes. Data outputs remain stable.
L	L	Data In	Write	Data on the data input bus D[n-1:0] is written to the memory location specified on the address bus A[m-1:0], and driven through to the data output bus Q[n-1:0].
L	H	SRAM Data	Read	Data on the data output bus Q[n-1:0] is read from the memory location specified on the address bus A[m-1:0].



Submission

- Create a folder named **studentID_hw3**, and put all below files into the folder
 - **core.v**
 - **core_syn.v**
 - **core_syn.sdf**
 - **core_syn.ddc**
 - **core_syn.area**
 - **core_syn.timing**
 - **all other design files** included in your file list (optional)
 - **report.txt**
 - **syn.tcl**
 - **rtl_01.f**
 - **rtl_03.f**
- Compress the folder **studentID_hw3** in a tar file named **studentID_hw3_vk.tar** (k is the number of version, $k = 1, 2, \dots$)
 - Use lower case for the letter in your student ID.
(Ex. r11943006_hw3_v1.tar)

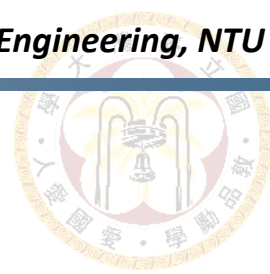


Grading Policy

- Correctness of simulation: **70%** (follow our spec)

Pattern	Description	RTL simulation	Gate-level simulation
tb0	Load + shift + scale + display	5%	5%
tb1	Load + shift + scale + conv.	5%	10%
tb2	Load + shift + median filter	5%	5%
tb3	Load + shift + Sobel + NMS	5%	10%
tb4	All operations (no display)	5%	5%
tbh	Hidden patterns	x	10%

- Performance: **30%**
 - Performance = **Area * Time** ($\mu\text{m}^2 * \text{ns}$)
 - Time = total simulation time of tb4**
 - The lower the value, the better the performance
 - Performance score only counts if your design passes all the test patterns

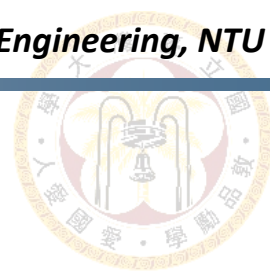


Grading Policy

- **No late submission**
 - 0 point for this homework

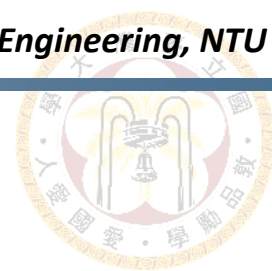
- Lose **5 points** for any wrong naming rule or format for submission
 - Do not directly compress all homework folders and upload it to NTU COOL
 - Make sure the code you upload can be decompressed and executed

- No plagiarism



Grading Policy

- **Violations of any spec (p.6 – p.9) incur point penalties**
 - Negative slack
 - 0 point for gate-level simulations and performance
 - Design without SRAM
 - 0 point for gate-level simulations and performance
 - Violate other rules but pass all simulations
 - Performance score * 0.7



Area

■ core_syn.area

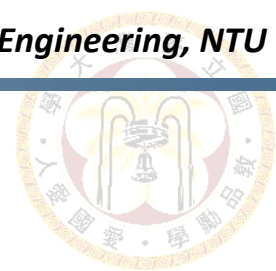
```
Number of ports:      883
Number of nets:       6074
Number of cells:      5142
Number of combinational cells: 4756
Number of sequential cells: 306
Number of macros/black boxes: 1
Number of buf/inv:    1241
Number of references: 269

Combinational area:   66609.370834
Buf/Inv area:         10982.178019
Noncombinational area: 12156.778545
Macro/Black Box area: 131906.968750
Net Interconnect area: 565696.185242

Total cell area:      210673.118129
Total area:           776369.303371
```

Number of macros/black boxes
should not be 0

210673.118129 μm^2

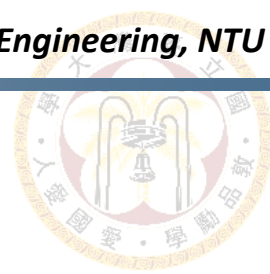


Report

- TA will run your design with your clock period
- **report.txt**

```
1  StudentID:  
2  
3  Clock period: 5.0 (ns)  
4  
5  Area : 210673.118129 (um^2)  
6
```

The clock period that can pass all gate-level simulations without any timing violations



References

- [1] Rounding to the nearest
 - [Rounding - MATLAB & Simulink \(mathworks.com\)](https://www.mathworks.com/help/matlab/matlab_prog/rounding-to-the-nearest.html)
- [2] R. C. Gonzalez and R. E. Woods, *Digital Image Processing*, 4th edition, Pearson, 2018.
- [3] Image gradients and Sobel kernels
 - [Image Gradients with OpenCV \(Sobel and Scharr\)](https://www.danielford.me/opencv/image-gradients-with-opencv-sobel-and-scharr/)